WEST

Help Logout Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences

Search Results -

Terms	Documents
l12 and ((707/8)!.CCLS.)	2

Database:
US Patents Full-Text Database
US Pre-Grant Publication Full-Text Database
JPO Abstracts Database
EPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Refine Search: Clear

Search History

Today's Date: 9/26/2001

DB Name	Query	Hit Count	Set Name
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l12 and ((707/8)!.CCLS.)	2	L29
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l12 and l27	0	L28
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	((707/103)!.CCLS.)	235	L27
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l12 and l19	3	L26
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l12 and l18	3	L25
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l12 and l17	3	L24
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l12 and l16	1	L23
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l12 and l15	6	L22
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l12 and l14	3	L21
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l12 and l13	8	L20
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	(((707/200)!.CCLS.))	693	L19
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	(((707/100)!.CCLS.))	828	L18
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	(((707/10)!.CCLS.))	1513	L17
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	(((707/5)!.CCLS.))	650	L16
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	(((707/3)!.CCLS.))	1377	L15
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	(((707/2)!.CCLS.))	899	L14
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	((707/1)!.CCLS.)	1157	L13
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l11 and secondary near storage	57	L12
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l8 and (primary near storage or main near storage)	147	L11
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l9 and database near manager	113	L10
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l8 and data near records	727	L9
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	database near manag$ near system	4121	L8
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l6 and second with storage	6	L7
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l5 and (primary with storage or main with storage)	14	L6
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l4 and database near manager	39	L5
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l3 and rank$	217	L4
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l2 and key	1712	L3
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l1 and data with record	2770	L2
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	database with manag$ with system	8870	L1

WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences

Search Results -

Terms	Documents
l12 and l19	3

Database:
US Patents Full-Text Database
US Pre-Grant Publication Full-Text Database
JPO Abstracts Database
EPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Refine Search: | Clear

Search History

Today's Date: 9/26/2001

DB Name	Query	Hit Count	Set Name
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l12 and l19	3	L26
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l12 and l18	3	L25
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l12 and l17	3	L24
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l12 and l16	1	L23
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l12 and l15	6	L22
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l12 and l14	3	L21
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l12 and l13	8	L20
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	(((707/200)!.CCLS.))	693	L19
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	(((707/100)!.CCLS.))	828	L18
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	(((707/10)!.CCLS.))	1513	L17
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	(((707/5)!.CCLS.))	650	L16
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	(((707/3)!.CCLS.))	1377	L15
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	(((707/2)!.CCLS.))	899	L14
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	((707/1)!.CCLS.)	1157	L13
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l11 and secondary near storage	57	L12
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l8 and (primary near storage or main near storage)	147	L11
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l9 and database near manager	113	L10
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l8 and data near records	727	L9
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	database near manag$ near system	4121	L8
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l6 and second with storage	6	L7
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l5 and (primary with storage or main with storage)	14	L6
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l4 and database near manager	39	L5
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l3 and rank$	217	L4
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l2 and key	1712	L3
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	l1 and data with record	2770	L2
USPT,PGPB,JPAB,EPAB,DWPI,TDBD	database with manag$ with system	8870	L1

WEST

Generate Collection

L20: Entry 4 of 8 File: USPT Jul 29, 1997

DOCUMENT-IDENTIFIER: US 5652873 A
TITLE: System and method for simulating a contiguous addressable data space

BSPR:
Computerized data processing systems rely on various types of storage spaces to process and store data. For example, "main storage" is program-addressable storage from which data can be loaded directly into the registers of the central processing unit (CPU) for processing. "Auxiliary storage" is addressable storage other than main storage that can be accessed by means of input/output (I/O) channels, and includes direct access storage devices such as magnetic storage disks. "Expanded storage" is a high-speed, high-volume electronic extension of main storage that is accessed synchronously in multiple-byte increments, e.g., 409 (4K) bytes, sometimes referred to as a "page".

BSPR:
In order for the CPU to process the data, the data normally should be in main storage. Main storage however, is limited and is therefore not used to store large amounts of data permanently. On the other hand, vast amounts of data may be stored on data disks. However, accessing data from disks is slow compared to the rate at which it can be processed in main storage. To compensate for the difference in access rates, a data buffer is used. A data buffer is a portion of storage used to hold input and output data temporarily. The data buffer can reside in main storage or expanded storage.

BSPR:
On multi-user computing systems, concurrent users time-share the resources on the computer systems through "virtual machines". In a virtual machine, which is a functional simulation of the real machine, each user addresses the computer main storage as though it were real. Addressable main storage in a virtual machine is called "virtual storage". The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of real disk storage available. Data in virtual storage is mapped to real addresses when the CPU references the data. Mapping is the establishment of correspondences between the physical storage and virtual storage locations. An address translation table is maintained by the operating system for this purpose.

BSPR:
On virtual machines, each user's reference to a memory address is referred to as a virtual address, and each range of addressable space available to a user is called an address space. When a user references a virtual storage location, the page containing that location may be on disk or expanded storage as indicated by a "flag" in the address translation table. When a page is to be copied to main storage, the operating system reads the page into an available real storage page location. When completed, the page translation table is updated to reflect the new page location. If no real storage space is available, the operating system frees up main storage space by "paging out" least recently used pages.

BSPR:
On some types of virtual machines, users access multiple address spaces. Some of these address spaces, however, contain only data (not computer instructions) and are referred to as data spaces. Furthermore, data space pages can be mapped to a data disk in such a manner as to eliminate the need for the database program manager to execute page I/O operations in order to move data between a data disk and main storage. On these systems, the location of a data object page on a data disk is known to the operating system. When the page is referenced by a user, it is read from its data disk location by the operating system without requiring a specific disk operation from the database program manager. When a page is directly accessible by the operating system, the database system operates more efficiently with less demands on CPU processing cycles.

BSPR:
In the prior art, various schemes are available to use data spaces. However, a method or means has not been found that discloses a multi-user system using data spaces in virtual memory for handling data objects of various sizes. Examples of prior art involving virtual memory but not addressing this deficiency, include: U.S. Pat. No. 4,742,447

("Method To Control I/O Processes In Multi-tasking Virtual Memory Virtual Machine Type Data Processing System") discloses a method for accessing information in a page segmented virtual memory data processing system in which virtual machines framing UNIX type operating systems are concurrently established, and in which a memory manager controls the transfer of information between primary storage and secondary storage devices in response to the occurrence of page faults.

BSPR:
U.S. Pat. No. 4,843,541 ("Logical Resource Partitioning of a Data Processing System") discloses a method and means for partitioning the resources in a data processing system into a plurality of logical partitions. The main storage, expanded storage, the channel and sub-channel resources of the system are assigned to the different logical partitions in the system to enable a plurality of preferred guest programming systems to run simultaneously in the different partitions.

DEPR:
The invention is implemented using the database storage system shown in FIG. 1. It comprises a directory disk 1, one or more data disks 2, and one or two log disks 2.1 similar to the data disks. The database storage system is part of a computer system having a central processing unit (CPU) 2.2 and memory 2.3. The CPU runs a database management system (DBMS) software program which manages the data stored in the storage devices 2.

DEPR:
Step 3. On first reference by a user to a page in the data object within the contiguous area 604, the invention provides for a mapping to disk only the segment 616a-616p of the data object containing the desired page. This partial mapping of the data object, rather than mapping the whole data object, minimizes the use of main storage because for each segment mapped, the operating system has to maintain a control block which uses up real storage. Delaying the mapping to a segment on the first reference means that a control block is only allocated if the segment 616a-616p is actually referenced. Therefore, data space segments 616a-616p that are never referenced, and thus not mapped, will not require a control block;

DEPR:
An example of mapping on demand, as provided by one aspect of the present invention, is illustrated in FIG. 14 and can be described using the IBM SQL/DS database management system operating on an IBM System/390 computer with the IBM VM/ESA operating system. Before updating a page, the IBM SQL/DS management system begins at step 1404 and copies the requested page to a buffer. It then reissues the MAPMDISK DEFINE request with the PAGEVIEW=ZERO parameter before copying the data back from the local buffer to the data space 608a-608q. Note, PAGEVIEW=ZERO informs VM/ESA operating system that it should provide an empty real storage page for that data space page 608a-608q rather than reading the contents of the disk 2 when the page is first referenced.

DEPR:
Partial save is illustrated in FIGS. 15-17 and can be described using the IBM SQL/DS database management system as follows:

CLPR:
1. A method for a database management system to simulate a database in a contiguous data space in computer memory, wherein the database comprises one or more data objects of variable size and is stored in one or more database storage disks, comprising:

CLPR:
3. A database management system for simulating a database in a contiguous data space in computer memory, wherein the database comprises one or more data objects of variable size and is stored in one or more database storage disks, comprising:

CLPR:
4. The database management system according to claim 3, wherein said incrementing means further comprises second incrementing means for incrementing said next available page identifier by a rounding number of pages, thereby said next available page identifier records a page number equal to a starting page boundary of a data segment in the contiguous data space.

CLPV:
(1) creating the contiguous data space by allocating and initializing a data space to sub-data space mapping table having a plurality of table entries, wherein each table entry includes a sub-data space identifier, wherein the contiguous data space is a concatenation of a plurality of sub-data spaces, each sub-data space being assigned a sub-data space identifier, wherein the contiguous data space and said sub-data spaces are addressable by the database management system with said sub-data space identifiers, and wherein the computer memory comprises a virtual memory and said sub-data spaces

comprise virtual data spaces;

CLPV:
creating means for creating the contiguous data space by allocating and initializing a data space to sub-data space mapping table having a plurality of table entries, wherein each table entry includes a sub-data space identifier, wherein the contiguous data space is a concatenation of a plurality of sub-data spaces, each sub-data space being assigned a sub-data space identifier, wherein the contiguous data space and said sub-data spaces are addressable by the database management system with said sub-data space identifiers and wherein the computer memory comprises a virtual memory and said sub-data spaces comprise virtual data spaces;

CCXR:
707/1

WEST

Generate Collection

L20: Entry 2 of 8 File: USPT Dec 1, 1998

DOCUMENT-IDENTIFIER: US 5845330 A
TITLE: Using an intermediate storage medium in a database management system

ABPL:
A database management system incorporating an intermediate level of storage medium. Software of the DBMS controls the transfer of data between a primary, secondary and intermediate storage mediums. This transfer is under control of the DBMS and is not transparent to the DBMS. The intermediate storage medium has an access time that is shorter than the access time of the secondary storage medium. During operation of the DBMS, data is read from the secondary storage medium and held in the primary storage medium until the storage medium is full. Thereafter, as additional data is read by the DBMS from the database stored on the secondary storage medium, data is transferred to the intermediate storage medium under control of the DBMS, thus preserving the benefit of having the data read into the primary storage medium.

BSPR:
This application relates to database management systems and, more particularly, to a database management system incorporating an intermediate storage medium.

BSPR:
Not all data processing systems allow a database management system to access the maximum amount of memory allowed by its hardware. For example, a high end server may have a 64 bit address length. Even though the hardware has a 64 bit address length and, thus, is theoretically capable of supporting an address space of up to 30 gigabytes, certain operating systems can support no more than 4 gigabytes for a single software application Moreover, sometimes there are not enough physical memory slots in the computer to allow the maximum amount of addressable memory.

BSPR:
Similarly, a low-end server may have a 32 bit address and, therefore, theoretically be capable of addressing up to 4 gigabytes of main memory. However, restrictions of the operating system or a lack of memory slots may mean that primary memory is actually limited to less than 1 gigabyte. The entire contents of a large database will not fit into the main memory of a conventional computer. Thus, databases are often stored on a secondary storage medium, such as a hard disk.

BSPR:
Conventional databases have long had to deal with the memory restrictions of the hardware on which they are executing. Conventional database management systems (DBMSs) bring portions of the database into primary storage/main memory from secondary storage when those portions are needed by the DBMS. If the primary storage cannot hold the entire database at once, however, the DBMS must manage the data, swapping it in and out of main memory as needed. This operation is called "paging."

BSPR:
Some computer systems employ the concept of hardware paging. Such hardware paging places a "fast" storage medium between a slow non-volatile secondary storage and the computer system. Whenever data is read from the secondary storage, the data is also placed in the fast storage medium. Thus, whenever the computer system performs a read operation of secondary storage, the fast storage medium is searched first to determine whether the desired information has previously been placed in the fast storage medium. The existence of the fast storage medium is completely transparent to the application software, which does not "know" that the fast storage medium exists and does not "know" when the data comes from the fast storage medium instead of from the secondary storage. Thus, the fast storage medium is not visible to software executing on the computer system. Such arrangements simply allow the system to deliver faster results than would otherwise be expected.

BSPR:
A problem involved with hardware paging is that once the fast storage medium is full, old data must be purged from the fast storage medium in order to make room for new data.

When old data that has been written to (i.e., "dirty" data) is purged, it is written to the larger secondary storage medium. Because the secondary storage medium is relatively slow, the benefit of previously bringing the data into the fast storage medium is lost for that data.

BSPR:
The present invention overcomes the problems and disadvantages of the prior art by introducing an intermediate level of storage medium in a database management system (DBMS). An embodiment of the present invention includes a large secondary storage medium, a smaller primary storage medium, and an intermediate storage medium that has a faster access time than the secondary storage medium. In a DBMS in accordance with the present invention, software in the DBMS controls the transfer of data between the primary, secondary, and intermediate storage mediums. This transfer is under control of the DBMS and is not transparent to the DBMS.

BSPR:
During operation of the DBMS, as data is read from the secondary storage medium, it is held in the primary storage medium until the primary storage medium is full. Thereafter, as additional data is read by the DBMS from the database stored on the secondary storage medium, data is transferred to the intermediate storage medium under control of the DBMS. This transfer makes room in the primary storage medium without completely losing the advantage of previously caching the data, since the intermediate storage medium is considerably faster than the secondary storage medium.

BSPR:
In accordance with the purpose of the invention, as embodied and broadly described herein, the invention is a method for accessing a database, comprising the steps, performed by a database management system stored in a secondary storage medium, of a data processing system, of: receiving a request to read database data located in the database; when a primary storage medium is full determining which database data needs to be discarded from the primary storage medium to make room for more data; storing the database data to be discarded into an intermediate storage medium; and reading the database data, after the storing step, into the primary storage medium from a one of the intermediate storage medium and the secondary storage medium.

BSPR:
In further accordance with the purpose of the invention, as embodied and broadly described herein, the invention is second an apparatus for accessing a database, comprising: a DBMS that manipulates data stored in the database; a primary storage medium; an intermediate storage medium having an access time; a secondary storage medium having a second access time that is longer than the first access time; and a data migration portion for, in accordance with a request from the DBMS, moving a piece of the data from the primary storage medium to the intermediate storage medium when the primary storage medium is full.

DRPR:
FIG. 4 shows an example of buffers used to store the database contents in a primary storage medium or an intermediate storage medium.

DRPR:
FIGS. 6(a) and 6(b) show respective examples of addressing schemes for the primary storage medium and for the intermediate storage medium.

DEPR:
FIG. 1 is a block diagram of a computer system 100 in accordance with a preferred embodiment of the present invention. Computer system 100 includes a processor 102; a primary storage medium 104; an intermediate storage medium 170; a secondary storage medium 180; an input device 150, such as a keyboard or mouse; and a display device 160, such as a display terminal. Primary storage medium 104, which includes buffer area 124, can include any type of computer memory including, without limitation, random access memory (RAM) and read only memory (ROM). Intermediate storage medium 170 can include, without limitation, an internal RAM disk or extended memory or any other appropriate type of storage internal to the primary storage medium. Accessing data from the primary storage medium 104 or the intermediate storage medium 170 by the DBMS may take, for example, 25 to 50 microseconds. Accessing data from the secondary storage medium 180 by the DBMS may take, for example, 8 to 10 milliseconds.

DEPR:
The described embodiment uses a 64 bit address and can support up to 30 gigabytes of primary storage. It is anticipated that future systems will have even larger memories. The invention can also, of course, operate in systems with smaller memories than that described herein The operating system of the described embodiment can address up to only 4 gigabytes of primary storage space per application, even though the hardware can support up to 30 gigabytes of address space. This extra memory space can be addressed

indirectly--for example, through installing an internal RAM disk or using an extended memory or other appropriate memory. Other sizes of address space may be used in conjunction with an internal RAM disk or extended memory.

DEPR:
Secondary storage medium 180 can include, without limitation, magnetic and optical storage medium such as magnetic or optical disks. Secondary storage 180 stores the data of the database. Certain portions of this data are loaded into primary storage medium 104 and intermediate storage medium 170, as described below.

DEPR:
Computer system 100 further includes an input device 161 for reading a computer usable medium 162 having computer readable program code means embodied therein. Input device 161 is, for example, a disk drive or CD ROM drive. The computer readable program means may be, for example, a DBMS 120. Primary storage medium 104 includes DBMS 120. Primary storage medium 104 also includes a page table 122 and memory area 124 (e.g., memory buffers) for storing data loaded from the database of intermediate/secondary storage 170/180.

DEPR:
A person of ordinary skill in the art will understand that primary storage medium 104 also contains additional information, such as application programs, operating systems, data, etc., which are not shown in the figure for the sake of clarity. It will be understood by a person of ordinary skill in the art that computer system 100 can also include numerous elements not shown in the figure for the sake of clarity, such as additional disk drives, keyboards, display devices, network connections, additional memory, additional processors, LANs, input/output lines, etc. A preferred embodiment of the invention runs under the Solaris operating system, Version 2.5.1. Solaris is a registered trademark of Sun Microsystems, Inc. Other implementations can execute in conjunction with other appropriate operating systems and execution environments.

DEPR:
FIGS. 2(a) and 2(b) are examples of pseudo-code showing steps performed by DBMS 120 to read data from the database of secondary storage medium 180 in accordance with the present invention. It will be understood that the pseudo-code of FIG. 2 is preferably embodied by computer instructions stored in a memory of system 100 and executed by processor 102. The computer instructions are preferably stored in a library accessed/linked by DBMS 120.

DEPR:
FIG. 2(a) shows pseudo-code for a read operation. In the example, DBMS 120 has decided to read a piece of DB information that it "knows" is stored in a certain page of the database (e.g., page X). In line 202, the DBMS searches page table 122 to determine whether Page X is present in primary storage buffer area 124. In line 204, if Page X is present in primary storage area 124 then, in line 206, the DBMS retrieves from the page table a location in primary storage area 124 where Page X is stored (e.g., a buffer location in area 124).

DEPR:
If Page X is not in primary storage area 124 then control passes to line 208. In line 208, the DBMS searches page table 122 to determine whether Page X is present in intermediate storage medium 170. In line 210, if Page X is present in intermediate storage medium 170 then, in line 212, the DBMS gets a free buffer in area 124, as described in connection with FIG. 2(b). In line 213, the DBMS retrieves from the page table a location in intermediate storage medium 170 where Page X is stored (e.g., a buffer location in intermediate storage medium 170). Examples of buffers in primary storage area 124 and in intermediate storage medium 170 are shown in FIG. 4. In the described embodiment, each storage medium 104 and 170 maintains a list of free(empty) buffers and a list of used buffers. Buffers are taken from the free list as they are needed. The DBMS copies the data from intermediate storage medium 170 to the empty buffer in primary storage area 124. In line 214, the DBMS returns the location in primary storage area 124 where Page X was placed (e.g., a buffer location in area 124) to the caller who requested the information.

DEPR:
If the data is not in the primary or intermediate storage media, then Page X is present in secondary storage medium 180. The DBMS proceeds to move the desired page from secondary storage medium 180 to a buffer in primary storage area 124. In line 216, the DBMS gets a free buffer in area 124, as described in connection with FIG. 2(b). In line 218, the DBMS does a physical read from secondary storage medium 180 into the free buffer in area 124. In step 220, the DBMS returns the buffer containing the results of the read to the caller who requested the information.

DEPR:

FIG. 2(b) shows pseudo-code for obtaining a free (empty) buffer in primary storage area 124. In lines 222 and 224, if there are free buffers in area 124, then one of these free buffers is returned to the calling routine. If there are not free buffers in area 124, in line 230, the DBMS finds a buffer in area 124 that contains data, but that has not been used for a long time. In a preferred embodiment, buffers are discarded in a "longest time since access" order, although any appropriate scheme could be used. In line 232, data in the buffer is moved from primary storage area 124 to intermediate storage medium 170. In line 234, the former contents of the buffer are cleared. This action leaves a free buffer in area 124. In line 236, the corresponding entry in page table 122 is cleared. The location of the free buffer in area 124 is returned to the calling routine.

DEPR:
Thus, in the described embodiment, data is read into primary storage medium 124 until that medium is full. Then, as additional data is read into primary storage medium, old data is moved from primary storage area 124 to intermediate storage medium 170 as needed to make room in primary storage area 124 for the new data. (Data may need to be purged from intermediate storage medium 170 to make room for the moved data). When a read operation is first performed, the DBMS looks for data in this order: primary storage area 124, intermediate storage medium 170, and secondary storage medium 180.

DEPR:
FIGS. 3(a) through 3(c) show an example of two successive reads from the database into primary storage medium 104/124. A first read requests Page X. A second read requests page Y. In FIG. 3(a), before X is read, page table 122 includes a page number field 302, a primary location field 304, an intermediate location field 306, and an age field 308. Other implementations can use two tables or could use other appropriate configurations for their page tables. In the example, before either of the read operations, primary storage medium has one free buffer 320. When the request to read Page X is received, the DBMS determines that Page X is not in the primary or intermediate storage media from examining the primary location field 304 and intermediate location field 306 of the page table. The DBMS then identifies the last free buffer in primary storage area 124 and copies Page X from secondary storage 180 to primary area 124. As shown in FIG. 3(b), the primary location field of the page table is adjusted to show that Page X is at this new location.

DEPR:
When the request to read Page Y is received, there are no more free buffers in primary storage area 124. Thus, an oldest buffer Z is moved from primary storage area 124 to intermediate storage medium 170 to make room for Page Y in primary storage area 124. The page table entries for Page Z and Page Y are altered to reflect the data movements.

DEPR:
In the described embodiment of the present invention, the DBMS always performs a write operation to secondary storage 180. Thus, page table 122 preferably contains a "dirty bit," indicating that the page has been written to (as is well-known). The write operation of the described embodiment is essentially the same as a conventional write operation.

DEPR:
FIG. 4 shows an example of buffers that can be used in primary storage area 124 and intermediate storage medium 170. In the described embodiment, each storage medium 104 and 170 maintains a list of free(empty) buffers and a list of used buffers. Buffers are taken from the free list as they are needed, in a manner known to persons of ordinary skill in the art.

DEPR:
The operating system of the described embodiment, however, can address up to only 4 gigabytes of primary storage space per application, even though the hardware can support up to 30 gigabytes of address space. Other sizes of address space may be used in conjunction with an external RAM disk or extended memory. Accessing data from the primary storage medium by the DBMS may take, for example, 25 to 50 microseconds. Accessing data from the intermediate storage medium by the DBMS may take, for example, 100 to 150 microseconds. Accessing data from the secondary storage medium by the DBMS may take, for example, 8 to 10 milliseconds.

DEPR:
In summary, the present invention incorporates an intermediate storage medium, such as an internal RAM disk, an external RAM disk, or an extended memory. During a read operation, when a primary storage medium is full, the DBMS moves data from the primary storage medium to the intermediate storage medium to make room in the primary storage medium for the newly read data Moving data to an intermediate storage medium instead of a slow secondary medium allows the system to maintain the benefits of having previously read data in the intermediate storage medium. The existence of the intermediate storage

medium is not transpare to the DBMS and migration of dat between the storage mediums 104, 170, and 180 are under control of the DBMS.

CLPR:
1. A method for accessing a database, comprising the steps, performed by a database management system stored in a secondary storage medium, of a data processing system, of:

CLPR:
3. The method of claim 2, wherein the intermediate storage medium has approximately the same access time as the primary storage medium.

CLPR:
4. The method of claim 2, wherein the memory capacity of the database is larger than the memory capacity of the primary storage medium.

CLPR:
7. The apparatus of claim 6, wherein the intermediate storage medium has approximately the same access time as the primary storage medium.

CLPR:
8. The apparatus of claim 6, wherein the memory capacity of the database is larger than the memory capacity of the primary storage medium.

CLPV:
when a primary storage medium is full, determining which database data needs to be discarded from the primary storage medium to make room for more data;

CLPV:
reading the database data, after the storing step, into the primary storage medium from one of the intermediate storage medium and the secondary storage medium.

CLPV:
determining whether the requested database data is stored in a primary storage medium;

CLPV:
when the requested database data is stored in the primary storage medium, reading the database data from the primary storage medium;

CLPV:
when the requested database data is not stored in the primary storage medium, determining whether the requested database data is stored in an intermediate storage medium;

CLPV:
when the requested database data is not stored in the intermediate storage medium, reading the database data from a secondary storage medium.

CLPV:
a primary storage medium;

CLPV:
a secondary storage medium having a second access time that is longer than the first access time; and

CLPV:
a data migration portion for, in accordance with a request from the DBMS, moving a piece of the data from the primary storage medium to the intermediate storage medium when the primary storage medium is full.

CLPW:
computer readable program code devices configured to cause a computer to effect, when a primary storage medium is full, determining which database data needs to be discarded from the primary storage medium to make room for more data;

CLPW:
computer readable program code devices configured to cause a computer to effect reading the database data, after the data to be discarded is stored, into the primary storage medium from a one of the intermediate storage medium and the secondary storage medium.

CCXR:
707/1